Class 06 HW

Hayoung A15531571

10/25/2021

#Section 1 : Improving analysis code by writing functions

QA. Can we improve this?

```
df <- data.frame(a=1:10, b=seq(200,400,length=10),c=11:20,d=NA)
df$a <- (df$a - min(df$a)) / (max(df$a) - min(df$a))
df$b <- (df$b - min(df$a)) / (max(df$b) - min(df$b))
df$c <- (df$c - min(df$c)) / (max(df$c) - min(df$c))
df$d <- (df$d - min(df$d)) / (max(df$a) - min(df$d))
```

We need to make our own function! For each set of data (a, b, c, d), they subtract the minimum value, and then divide that by the maximum - minimum value of that set.

Let's test it on a small piece of data!

```
y <- c(1,2,3,4,5)
(y - min(y))/(max(y)-min(y))
```

```
## [1] 0.00 0.25 0.50 0.75 1.00
```

nice! but! we also have NA as part of our values

```
y <- c(1,2,3,4,5, NA)

#so let us make NA read as a numeric, as well as become 0
y <- as.numeric(y)
y[which(is.na(y))] = 0

(y - min(y))/(max(y)-min(y))
```

```
## [1] 0.2 0.4 0.6 0.8 1.0 0.0
```

PERFECT. okay now let's try to apply it to our problem.

```
#let's make our variable x the data frame
x <- unlist(df)

#first let's make all the data numeric, since there is NA involved
x <- as.numeric(x)
```

```r
#next we will map all the NA values as zero
x[which(is.na(x))] = 0

#let's plug in x (each set) into our wanted equation
(x - min(x))/(max(x)-min(x))
```

```
##  [1] 0.00000000 0.05555556 0.11111111 0.16666667 0.22222222 0.27777778
##  [7] 0.33333333 0.38888889 0.44444444 0.50000000 0.50000000 0.55555556
## [13] 0.61111111 0.66666667 0.72222222 0.77777778 0.83333333 0.88888889
## [19] 0.94444444 1.00000000 0.00000000 0.05555556 0.11111111 0.16666667
## [25] 0.22222222 0.27777778 0.33333333 0.38888889 0.44444444 0.50000000
## [31] 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
## [37] 0.00000000 0.00000000 0.00000000 0.00000000
```

we get the correct answers all at once! lets try making this into a function

```r
x <- df

output <- function(x) {
  x <- as.numeric(x)
  x[which(is.na(x))] = 0
  (x - min(x))/(max(x)-min(x))
}

#let's try it out

output(x[,1])
```

```
##  [1] 0.0000000 0.1111111 0.2222222 0.3333333 0.4444444 0.5555556 0.6666667
##  [8] 0.7777778 0.8888889 1.0000000
```

```r
#IT WORKS. we simply put in the column number to find the values!

output(x[,2])
```

```
##  [1] 0.0000000 0.1111111 0.2222222 0.3333333 0.4444444 0.5555556 0.6666667
##  [8] 0.7777778 0.8888889 1.0000000
```

```r
output(x[,3])
```

```
##  [1] 0.0000000 0.1111111 0.2222222 0.3333333 0.4444444 0.5555556 0.6666667
##  [8] 0.7777778 0.8888889 1.0000000
```

```r
output(x[,4])
```

```
##  [1] NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN
```

#Let's move on to the actual hw for now....

```
# Can you improve this analysis code?
library(bio3d)
s1 <- read.pdb("4AKE") # kinase with drug
```

```
##   Note: Accessing on-line PDB file
```

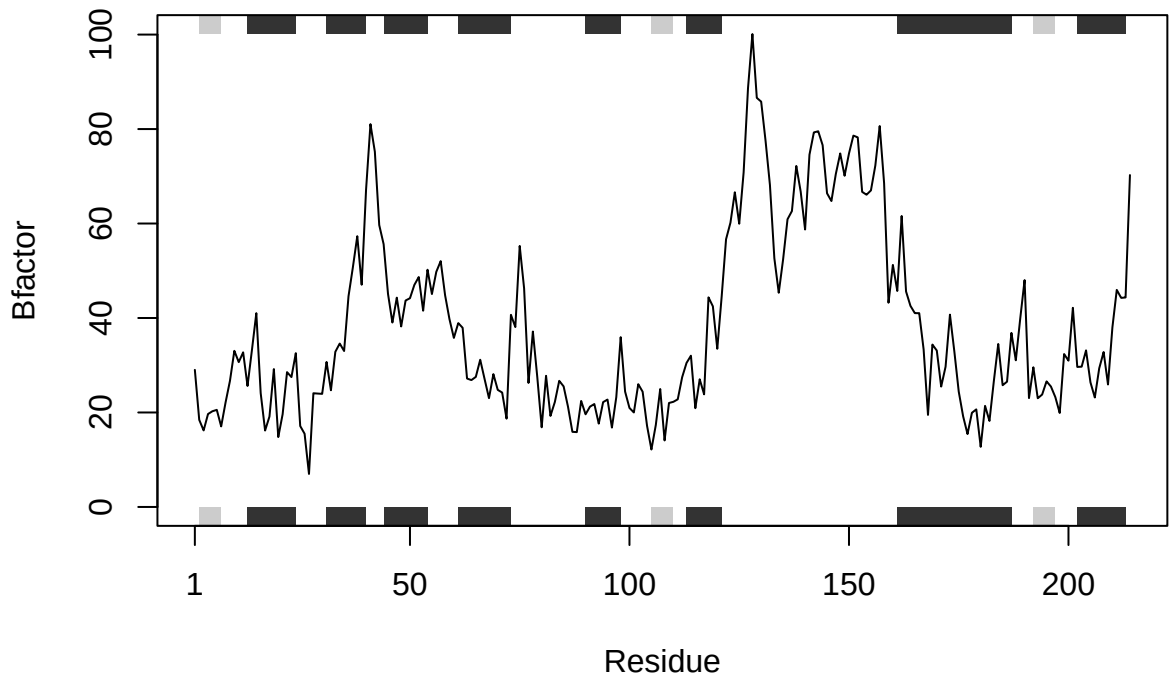
```
s2 <- read.pdb("1AKE") # kinase no drug
```

```
##   Note: Accessing on-line PDB file
##    PDB has ALT records, taking A only, rm.alt=TRUE
```

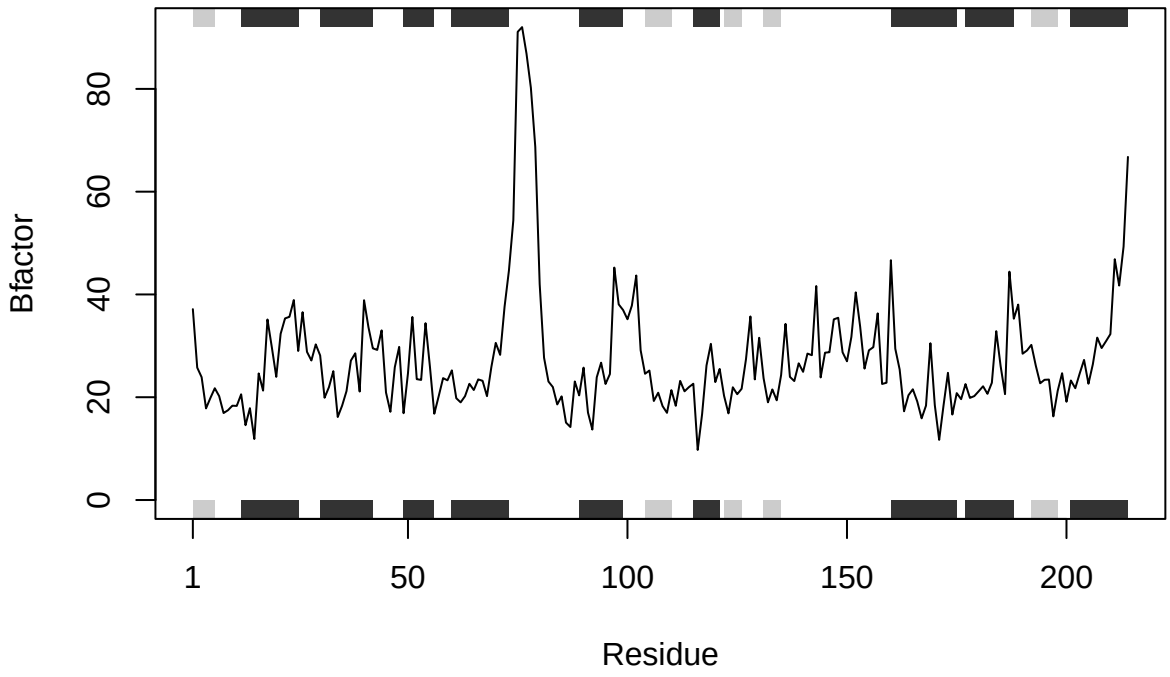
```
s3 <- read.pdb("1E4Y") # kinase with drug
```

```
##   Note: Accessing on-line PDB file
```

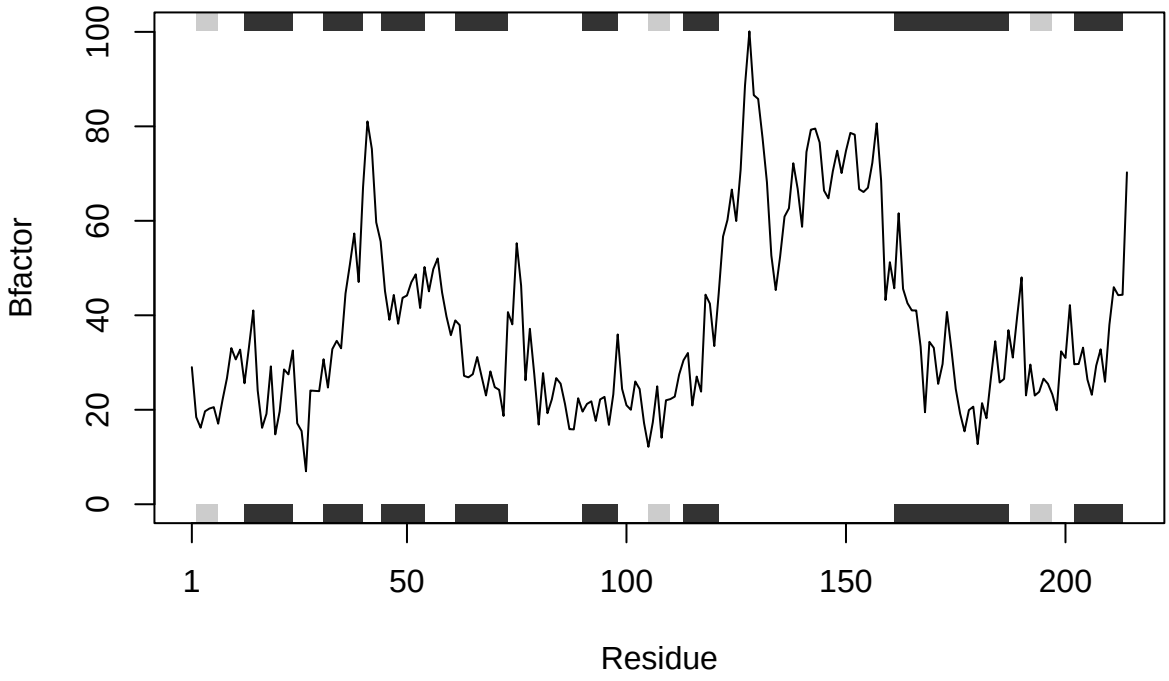
```
s1.chainA <- trim.pdb(s1, chain="A", elety="CA")
s2.chainA <- trim.pdb(s2, chain="A", elety="CA")
s3.chainA <- trim.pdb(s1, chain="A", elety="CA")
s1.b <- s1.chainA$atom$b
s2.b <- s2.chainA$atom$b
s3.b <- s3.chainA$atom$b
plotb3(s1.b, sse=s1.chainA, typ="l", ylab="Bfactor")
```



```
plotb3(s2.b, sse=s2.chainA, typ="l", ylab="Bfactor")
```




```
plotb3(s3.b, sse=s3.chainA, typ="l", ylab="Bfactor")
```



Q1. What type of object is returned from the read.pdb() function?

By using the read.pdb() command, R is accessing a PDB file with information on atoms, strucutre and sometimes more information. We are accessing this information, and R will now return a PDB structure object with the specific atom/file that we are reading.

Q2. What type of object is returned from the read.pdb() function?

trim.pdb() is used to create a new PDB object that is specifically based on our selection of backboneatoms. It is a smaller, cleaner PDB object.

Q3. What input parameter would turn off the marginal black and grey rectangles in the plots and what do they represent in this case?

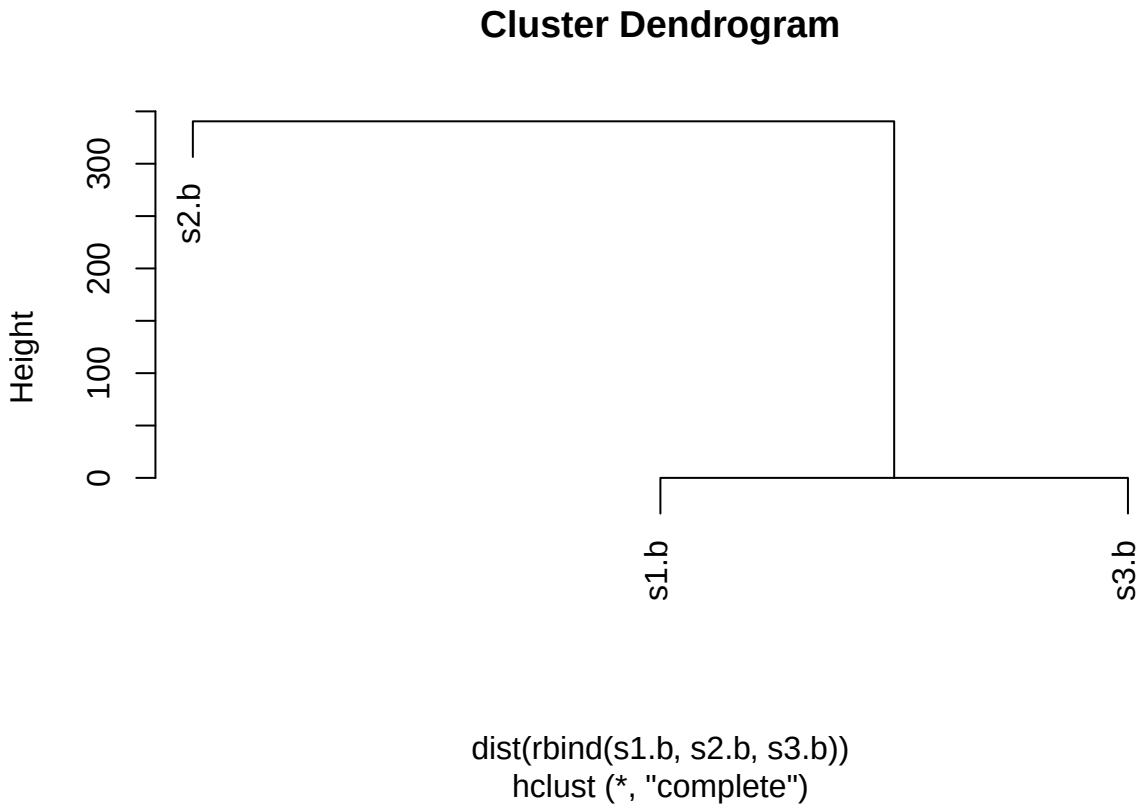
The black and grey rectangles are from the line with plotb3(), they represent pre-residue numeric vectors for a given protein structure, and are a schematic representation of major secondary structure elements. This is the classic version.

Q4. What would be a better plot to compare across the different proteins?

If we want to compare the structures of the proteins, it would be better to plot the 3D structure of the proteins superimposed on one another so that we can easily see the differences visually. We can do this using pdbfit()

Q5. Which proteins are more similar to each other in their B-factor trends. How could you quantify this?

```
hc <- hclust( dist( rbind(s1.b, s2.b, s3.b) ) )
plot(hc)
```



s1 and s3 are most similar in their B-factor trends. By first observing their plots we see that the trends are very similar, in addition the x-axis and y-axis ranges are also the same. Finally the code that we used above shows that s1 and s3 are most similar to one another.

Q6. How would you generalize the original code above to work with any set of input protein structures?

```
#this loads bio3d, only have to do it once so I won't include it in the function itself
library(bio3d)

#This makes the variables that will be read, the repeats in our function
s1 <- read.pdb("4AKE") # kinase with drug
```

```
##   Note: Accessing on-line PDB file
```

```
## Warning in get.pdb(file, path = tempdir(), verbose = FALSE): /var/folders/9_/
## 8mn2s75d3ql9q6sppw500jmr0000gn/T//Rtmps1qGy0/4AKE.pdb exists. Skipping download
```

```
s2 <- read.pdb("1AKE") # kinase no drug
```

```
##   Note: Accessing on-line PDB file
```

```
## Warning in get.pdb(file, path = tempdir(), verbose = FALSE): /var/folders/9_/
## 8mn2s75d3ql9q6sppw500jmr0000gn/T//Rtmps1qGy0/1AKE.pdb exists. Skipping download
```

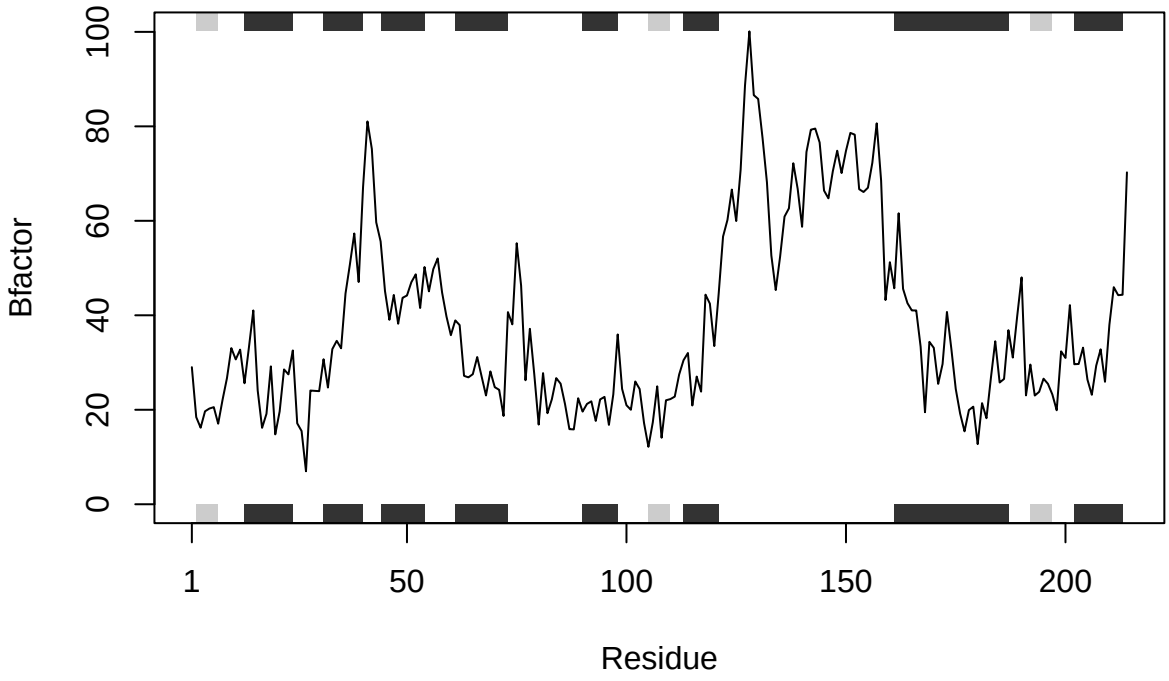
```
##    PDB has ALT records, taking A only, rm.alt=TRUE
```

```
s3 <- read.pdb("1E4Y") # kinase with drug
```

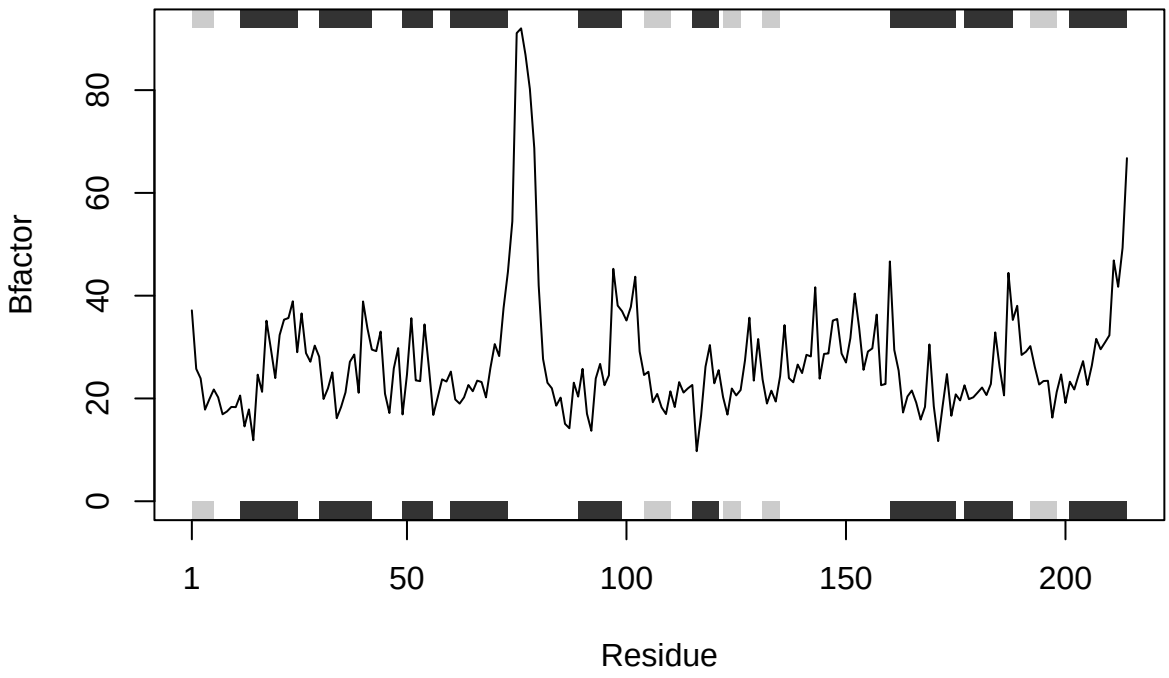
```
##   Note: Accessing on-line PDB file
```

```
## Warning in get.pdb(file, path = tempdir(), verbose = FALSE): /var/folders/9_/
## 8mn2s75d3ql9q6sppw500jmr0000gn/T//Rtmps1qGy0/1E4Y.pdb exists. Skipping download
```

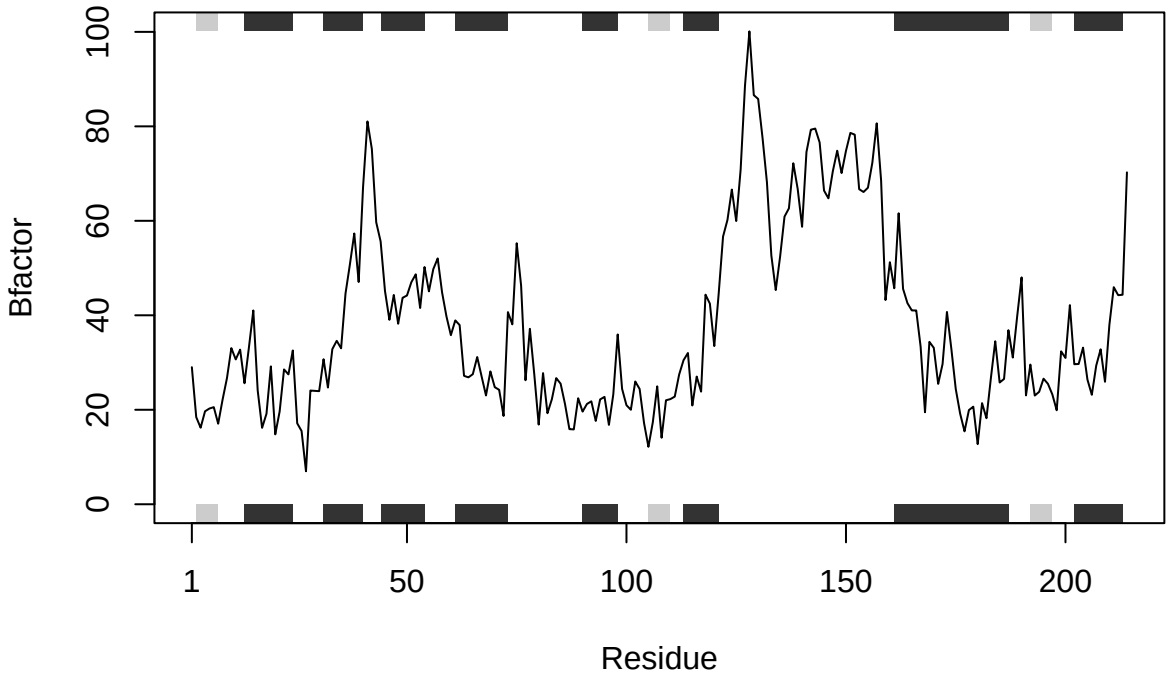
```
#this creates new objects
s1.chainA <- trim.pdb(s1, chain="A", elety="CA")
s2.chainA <- trim.pdb(s2, chain="A", elety="CA")
s3.chainA <- trim.pdb(s1, chain="A", elety="CA")

#Must do this for every "s_"
s1.b <- s1.chainA$atom$b
s2.b <- s2.chainA$atom$b
s3.b <- s3.chainA$atom$b

#finally create the plot
plotb3(s1.b, sse=s1.chainA, typ="l", ylab="Bfactor")
```




```
plotb3(s2.b, sse=s2.chainA, typ="l", ylab="Bfactor")
```




```
plotb3(s3.b, sse=s3.chainA, typ="l", ylab="Bfactor")
```




```
#i found a guide online that helps!
general <- function(file, chain, elmnt, fctr) {
  #adding colors after testing
    plot_colors <- c("red", "blue", "black")

  #to read the files first THIS IS FROM DATACAMP
  for (i in 1:length(file)) {
  s1 <- read.pdb(file[i])

  #create new objects
   s1.chain <- trim.pdb(s1, chain = chain, elety = elmnt)
   atom_df <- s1.chain$atom

  #do the $ part, but we are trying to make it more general now, so remove that and replace with fctr
   s1.fctr <- atom_df[, fctr]

  # PLOT TIME

  if (i == 1) {
    plotb3(s1.fctr, sse = s1.chain, typ = "l", ylab = paste(toupper(fctr), "factor", sep = ""), col = pl

    #adds additional plots on top of each other! (found this online)
  } else {
    lines(s1.fctr, col = plot_colors[i])
  }
```

```
    #must add legend
    legend("topright", title = "PDB File Name", file, fill = plot_colors, horiz=TRUE, cex = 0.5, inset =
}

}
```

NOW TO TEST IT

```
#read it first
"4AKE" <- read.pdb("4AKE") # kinase with drug
```

```
##   Note: Accessing on-line PDB file
```

```
## Warning in get.pdb(file, path = tempdir(), verbose = FALSE): /var/folders/9_/
## 8mn2s75d3ql9q6sppw500jmr0000gn/T//Rtmps1qGy0/4AKE.pdb exists. Skipping download
```

```
"1AKE" <- read.pdb("1AKE") # kinase no drug
```

```
##   Note: Accessing on-line PDB file
```

```
## Warning in get.pdb(file, path = tempdir(), verbose = FALSE): /var/folders/9_/
## 8mn2s75d3ql9q6sppw500jmr0000gn/T//Rtmps1qGy0/1AKE.pdb exists. Skipping download
```

```
##    PDB has ALT records, taking A only, rm.alt=TRUE
```

```
"1E4Y" <- read.pdb("1E4Y") # kinase with drug
```

```
##   Note: Accessing on-line PDB file
```

```
## Warning in get.pdb(file, path = tempdir(), verbose = FALSE): /var/folders/9_/
## 8mn2s75d3ql9q6sppw500jmr0000gn/T//Rtmps1qGy0/1E4Y.pdb exists. Skipping download
```

```
files <- c("4AKE", "1AKE", "1E4Y")
chains <- "A"
elements <- "CA"
factors <- "b"

general(files, chains, elements, factors)
```

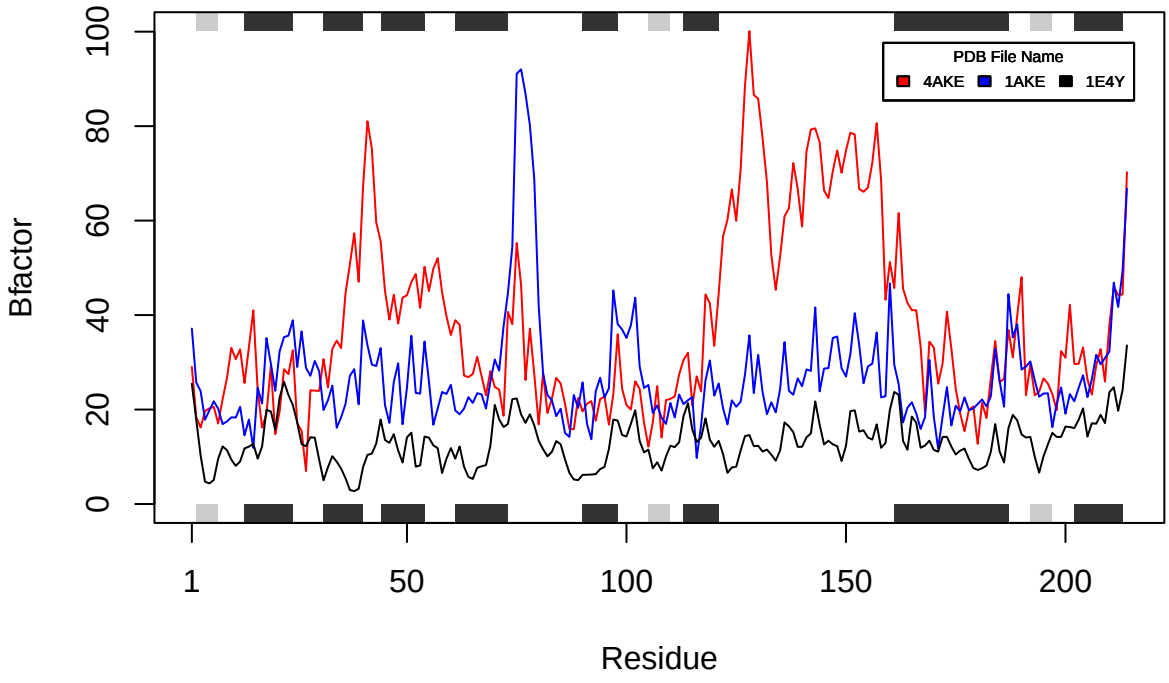
```
##   Note: Accessing on-line PDB file
```

```
## Warning in get.pdb(file, path = tempdir(), verbose = FALSE): /var/folders/9_/
## 8mn2s75d3ql9q6sppw500jmr0000gn/T//Rtmps1qGy0/4AKE.pdb exists. Skipping download
```

```
##   Note: Accessing on-line PDB file
```

```
## Warning in get.pdb(file, path = tempdir(), verbose = FALSE): /var/folders/9_/
## 8mn2s75d3ql9q6sppw500jmr0000gn/T//Rtmps1qGy0/1AKE.pdb exists. Skipping download
```

```
##    PDB has ALT records, taking A only, rm.alt=TRUE
##   Note: Accessing on-line PDB file

## Warning in get.pdb(file, path = tempdir(), verbose = FALSE): /var/folders/9_/
## 8mn2s75d3ql9q6sppw500jmr0000gn/T//Rtmps1qGy0/1E4Y.pdb exists. Skipping download
```



I get an error that plot colors were not found, so I'll add that into the function!

IT WORKED